Your scripts are awesome, they deserve their own EventLog

Something about EventLogs and easily creating your own one

Agenda

- History – how did all come together
- Basics about Windows EventLog Plattform
- PowerShell module WinEventLogCustomization
- Q&A

Get-Person -Short

Andreas Bellstedt

 @AndiBellstedt

 github.com/AndiBellstedt

IT infrastructure guy and PowerShell fellow

Microsoft Infrastructure

Security

The story behind the module

Once upon a time…

How does it happen

- First of all
  ....all you see is a result of curiosity

- EventLogs were there forever
  - Every component and a lot of software use them
  - Practically good for recognizing errors in the system

- Do you know Windows Event Forwarding?
- Have you ever heard about Project Sauron?
  - https://github.com/russelltomkins/Project-Sauron
  - Gets my interest back in 2017-2018

- Decision to put a playbook script into a utility
- A new module was born:
  WinEventLogCustomization

WinEventLogCustomization

Plattform	Information
PowerShell gallery	psgallery v1.0.0 platform windows downloads 13
GitHub	release v1.0.0.0 license MIT open issues 0 last commit: master july last commit: development july

Description

A PowerShell module helping you build custom eventlog channels and registering them into Windows Event Viewer. The build logs appear under "Application and Services", even like the "Windows PowerShell" or the "PowerShellCore/Operational" EventLog.

All cmdlets are build with

- powershell regular verbs
- pipeline availabilities wherever it makes sense
- comprehensive logging on verbose and debug channel by the logging system of PSFramework

Prerequisites

- Windows PowerShell 5.1
- PowerShell 6 or 7
- Administrative Priviledges are required for registering or unregistering EventChannels

Parts of project Sauron

Custom EventLog Channels

Event Forwarding Definitions

Module
WinEventLogCustomization

GPO to enable EventForwarding

Module
WindowsEventForwarding

Windows EventLog Plattform

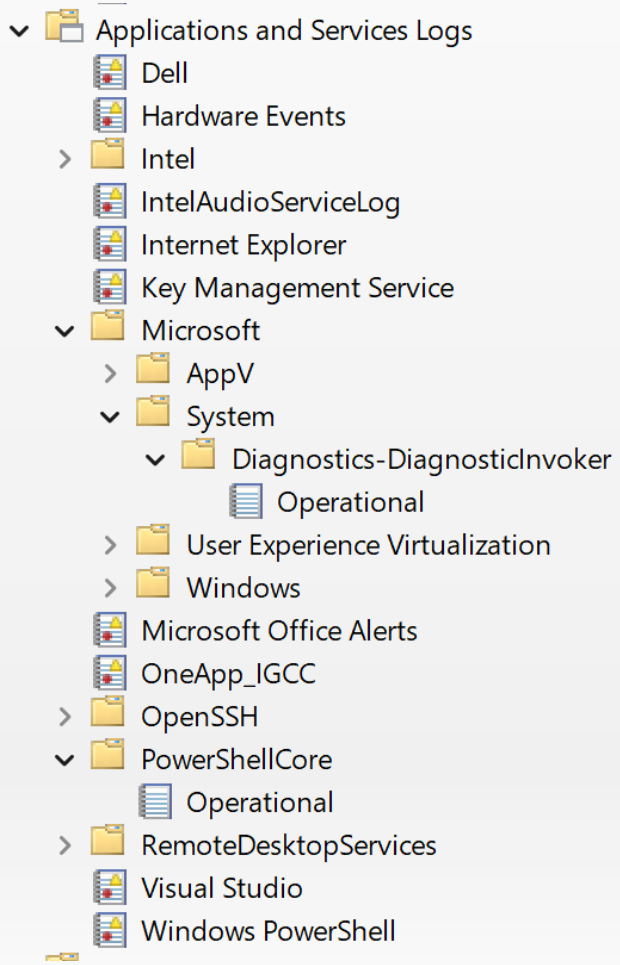
Some basics about EventLogs

and

customization

Windows Logs vs Applications & Services Logs

- Classic Logs
  - Has been there forever
  - General available
  - Flat structure
  - What a mess
- Applications & Services Logs
  - Specialized
  - Granularity
  - Hierarchical – depth of 1, 2 or 4



EventLog Channels

- File format EVT / EVTX
- Default is folder
  - %SystemRoot%\System32\Winevt\Logs\
  - But they can be anywhere
- Registration in Eventviewer with manifest / dll file

Creating custom EventLogs

*may contain traces of sarcasm

- No complex procedure!*
- Start with downloading the Windows

Create Manifest file for your Channel

*may contain traces of sarcasm

- Know the limits of the platform (no more then 8 channels in 1 provider)
- You can use the the (Event Channel) Manifest Generator
  - ecmangen.exe- from SDK
- Otherwise, you can write it on your own, it's plain text… just become comfortable with the eventman xml definition

```
You, 42 seconds ago | 1 author (You) | eventman.xsd (xsi:schemaLocation)
<?xml version="1.0"?>
<instrumentationManifest xsi:schemaLocation="http://schemas.microsoft.com/win/2004/08/events eventman.xsd" xmlns="http://schemas.microsoft.com/win/2004/08/events" xmlns:w
    <instrumentation>
        <events>
            <provider name="My-Own-Channel" guid="{CF27F07F-7013-483A-BC74-97A0F6AA32FC}" symbol="My_Own_Channel" resourceFileName="C:\Windows\system32\MyOwnChannel.dll"
                <channels>
                    <channel name="My-Own-Channel/Information" chid="My-Own-Channel/Information" symbol="My_Own_Channel_Information" type="Admin" enabled="false" />
                </channels>
            </provider>
        </events>
    </instrumentation>
</instrumentationManifest>
```

Compile the manifest

- Use with mc.exe from SDK to generate rc & h file

```
"C:\Program Files (x86)\Windows Kits\8.1\bin\x64\mc.exe" C:\ECMan\MyOwnChannel.man
```

- Use mc.exe from SDK again to generate cs file

```
"C:\Program Files (x86)\Windows Kits\8.1\bin\x64\mc.exe" -css MyOwnChannel.DummyEvent
C:\ECMan\MyOwnChannel.man
```

- Use rc.exe from SDK to generate a res file

```
"C:\Program Files (x86)\Windows Kits\8.1\bin\x64\rc.exe" C:\ECMan\MyOwnChannel.rc
```

- Use csc.exe from local .NET Framework within Windows to compile the dll from cs & res file

```
"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe" /win32res:C:\ECMan\MyOwnChannel.res
/unsafe /target:library /out:C:\ECMan\MyOwnChannel.dll C:\ECMan\MyOwnChannel.cs
```

Registering custom EventLog channel

*may contain traces of sarcasm

- That's all you can load the manifest up, now

```
wevtutil im c:\Windows\system32\MyOwnChannel.man
```

- Just keep in mind to fulfill the paths, you've specified in the manifest file, or at least, keep the man next to the dll file*

- ...and enable the channel
- ...and set properties on the channel

```
wevtutil sl My-Own-Channel/Information /lfn:E:\Logs\MyOwnChannel.evtx
wevtutil sl My-Own-Channel/Information /ms:1073741824
```

Finished creating custom EventLog channel

- That's it!
- You did it!
- Congratulation!

- You've followed a documentation from a Microsoft blog post
  - https://docs.microsoft.com/de-de/archive/blogs/russellt/creating-custom-windows-event-forwarding-logs


SAY
WHAAAAAT
memes.com

Introducing the module

A more enjoyable way

PowerShell module WinEventLogCustomization

- Ease the pain of creating custom EventLog channels
- Create & compile manifest & dll file in one command
  - For one or more channels
  - From interactive input or from a Excel template
- Register or Unregister Channels
- Set configuration on channels

Minor but recognizable things

- The functions from the module can move manifest/dll-file to different folders without breaking the definitions
  - Takes care of name and paths withing the xml structure of the module

- Can operate remotely
  - PSRemoting is used, so for obvious reason, Remoting has to be enabled

PowerShell style way of working

- Commands have rich pipeline support
  - Get | Set behavior
  - Various parametersets
  - WhatIf possibility and confirmation on changing or removing commands
- Parameters are documented
- Examples are inside the commands

A look inside the module

Bricks
and concrete

The way it was built

- Project template from PowershellFrameworkCollective module “PSModuleDevelopment”
- Using PSFramework for great logging and remote execution
- Create classes in a little C# project
  - No more than the struct of the objects
  - Easy the pipeling handling
- Type- and Format ps1xml definition file
  - Comfortable output in the console
  - Add calculated fields to the objects

The little helpers

- Use public and free "EPPlus.Net40.dll" from JanKallman to access excel files without present Installation of Excel
  - https://github.com/JanKallman/EPPlus
- Adopt the import function from module ImportExcel of Doug Finke
  - https://github.com/dfinke/ImportExcel

Finally

Conclusion

- Projects from curiosity don‘t go the easy way
- Curiosity brings knowledge
- Module is built with
  - Comfortable functions in mind
  - Verbose logging to tell what is going on under the hood
- Module WinEventLogCustomization is on the PowerShell Gallery

Question & Answers

Andreas Bellstedt

@AndiBellstedt

github.com/AndiBellstedt

More Information & Links

- Project Sauron
  - https://github.com/russelltomkins/Project-Sauron
  - https://docs.microsoft.com/de-de/archive/blogs/russellt/project-sauron-introduction
- Custom EventLogs
  - https://docs.microsoft.com/de-de/archive/blogs/russellt/creating-custom-windows-event-forwarding-logs
  - https://github.com/nsacyber/Event-Forwarding-Guidance
- Related Topic
  - WindowsEventForwarding PowerShell Module - https://github.com/PSSecTools/WindowsEventForwarding
  - Import-Excel - https://github.com/dfinke/ImportExcel
  - Working with Excel files - https://github.com/JanKallman/EPPlus

– Thank you –

Andreas Bellstedt

 @AndiBellstedt

 github.com/AndiBellstedt